VBA: Transfer Milepost M Coordinates From Route To Features

Contributed by Bert Granberg
06, May. 2009
Last Updated 06, May. 2009

One of the Interfaces in ArcObjects, IHitTest , is extremely useful for exploring proximity ('nearest') relationships between two features. It's very fast and doesn't much care what types of geometry you throw at it.

The code below shows an example of how IHitTest can be used to transfer milepost measure or m coordinates from a route feature (think all of I-15NB as one route feature) to range attributes of all the street features that make up the route. Route features are great for doing linear referencing but a lot of applications can do 'geocoding' style route-milepost location better if the milepost values at each end of the street feature are stored the same way address ranges are stored.

Basically, the code looks at each street feature that should get a milepost range (based on whether it has a DOT_RTNAME value or not) and then queries the route feature class to find the corresponding 'parent' route. Then it takes each end point for the street feature and queries the parent route for the nearest vertex to the specified point. Lastly, the code gets the m coordinate stored at that vertex to throw into a range field in the street feature class.

Street centerline features in the new SGID 9.3 SDE database will carry From and To milepost approximations on UDOT state and federal routes.

```
Public Sub milepostRangeLookup()

    Dim pMxDoc As IMxDocument
    Dim pMap As IMap
    Set pMxDoc = ThisDocument
    Set pMap = pMxDoc.FocusMap

    Dim pSrcLayer As IFeatureLayer
    Dim pSrcFeatureClass As IFeatureClass
    Dim pSrcFeatureCursor As IFeatureCursor
    Dim pSrcFeature As IFeature
    Dim pTarPolyline As IPolyline
    Dim pTarEndPt As IPoint
    Dim pTarTopOp As ITopologicalOperator
    Dim pTarPolygon As IPolygon
    Dim tarRtNameID As String
    Dim pTarLayer As IFeatureLayer
    Dim pTarFeatureClass As IFeatureClass
    Dim pTarFeatureCursor As IFeatureCursor
    Dim pTarFeature As IFeature
    Dim pTarDataset As IDataset
    Dim pTarWS As IWorkspace

    Dim pHitTest As IHitTest
    Dim hitStart As Boolean
    Dim hitend As Boolean
    Dim pHit As IPoint
    Dim dDist As Double
    Dim lPartIndex As Long
    Dim lSegIndex As Long
    Dim bRight As Boolean

    Dim pQF As IQueryFilter
    Dim pQF2 As IQueryFilter
    Dim pSrcRt As IPolyline
    Dim pSRcRTPC As IPointCollection
    Dim pTO As ITopologicalOperator

    Dim tarDOTMPFromFI As Integer
    Dim tarDOTMPToFI As Integer
    Dim tarDOTRTNameFI As Integer
    Dim Count As Long
```

```
Set pQF = New QueryFilter
pQF.WhereClause = "(not DOT_RTNAME is null) and DOT_RTNAME <> ""

Set pSrcLayer = pMap.Layer(1)
Set pSrcFeatureClass = pSrcLayer.FeatureClass

Set pTarLayer = pMap.Layer(0)
Set pTarFeatureClass = pTarLayer.FeatureClass
Set pTarDataset = pTarFeatureClass
Set pTarWS = pTarDataset.Workspace
tarDOTMPFromFI = pTarFeatureClass.FindField("DOT_F_MP")
tarDOTMPToFI = pTarFeatureClass.FindField("DOT_T_MP")
tarDOTRTNameFI = pTarFeatureClass.FindField("DOT_RTNAME")

Dim pEditor As IEditor
Dim pUID As New UID
pUID = "esriEditor.Editor"
Set pEditor = Application.FindExtensionByCLSID(pUID)

If pEditor.EditState = esriStateEditing Then
   MsgBox "You must not currently be in an edit session to run this script.", vbOKOnly, "Exiting"
   Exit Sub
End If

pEditor.StartEditing pTarWS
pEditor.StartOperation
On Error GoTo errorhandler

Set pTarFeatureCursor = pTarFeatureClass.Update(pQF, True)
Set pTarFeature = pTarFeatureCursor.NextFeature

Do Until pTarFeature Is Nothing

   Count = Count + 1

   Set pQF2 = New QueryFilter
   pQF2.WhereClause = "LABEL = '" & pTarFeature.Value(tarDOTRTNameFI) & "'"

   Set pSrcFeatureCursor = pSrcLayer.Search(pQF2, True)
   Set pSrcFeature = pSrcFeatureCursor.NextFeature

   If Not pSrcFeature Is Nothing Then


      'do for startpoint
      Set pTarPolyline = pTarFeature.Shape
      Set pSrcRt = pSrcFeature.Shape
      Set pTarEndPt = pTarPolyline.FromPoint

      dDist = 0
      lPartIndex = 0
      lSegIndex = 0
      bRight = False
      hitStart = False
      Set pHit = New Point
      Set pHitTest = pSrcRt
      hitStart = pHitTest.HitTest(pTarEndPt, 2, esriGeometryPartVertex, pHit, dDist, lPartIndex, lSegIndex, bRight)
      'Debug.Print "Hit point X, Y : " & pHit.X & " , " & pHit.Y
      'Debug.Print "Hit Distance : " & dDist
      'Debug.Print "Hit M Coordinate : " & CStr(CLng(pHit.M * 1000) / 1000)

      If hitStart Then
         pTarFeature.Value(tarDOTMPFromFI) = CStr(CLng(pHit.M * 1000) / 1000)
         Else
         Debug.Print "start not found: " & pTarFeature.OID & " " & pTarFeature.Value(tarDOTRTNameFI)
```

```
        End If


        'do for endpoint
        Set pTarEndPt = pTarPolyline.ToPoint
        dDist = 0
        lPartIndex = 0
        lSegIndex = 0
        bRight = False
        hitend = False
        Set pHit = New Point
        Set pHitTest = pSrcRt
        hitend = pHitTest.HitTest(pTarEndPt, 2, esriGeometryPartVertex, pHit, dDist, lPartIndex, lSegIndex, bRight)
        If hitend Then
           pTarFeature.Value(tarDOTMPToFI) = CStr(CLng(pHit.M * 1000) / 1000)
        Else
           Debug.Print "   end not found: " & pTarFeature.OID & " " & pTarFeature.Value(tarDOTRTNameFI)
        End If


        If hitStart Or hitend Then
           pTarFeatureCursor.UpdateFeature pTarFeature
        End If

      Else
        Debug.Print "*** no rt found for " & pTarFeature.OID & " " & pTarFeature.Value(tarDOTRTNameFI)
      End If

      'Debug.Print Count

      Set pTarFeature = pTarFeatureCursor.NextFeature

   Loop

   pEditor.StopOperation "Transfer Milepost Ranges"
   pEditor.StopEditing (True)
   Exit Sub
errorhandler:
   Debug.Print "Error on: " & pTarFeature.OID
   pEditor.StopOperation "Transfer Milepost Ranges"
   pEditor.StopEditing (False)

End Sub
```